Physics Parallelization

Erwin Coumans
SCEA
Takeaway

- Multi-threading the physics pipeline
  - collision detection and constraint solver
- Refactoring algorithms and data for SPUs
- Maintain unified multi-platform codebase
  - SPU task maps well to multi-core thread
Parallelism and SPUs

- Both data and code needs to fit in 256kb
- DMA transfer to upload code and data
  - Similar to GPU shader and texture upload
- Hide latency using double buffering
  - Upload next batch while processing current
- Patterns for Parallel Programming [TM2005]
Broadphase N-Body

- Find potential overlapping pairs
- $O(n^2)$ number of pairs in theory
- But $O(n)$ in practice
Broadphase Methods

- Uniform Grid
- Octree, BSP tree
- Spatial Hashing
  - Multi-resolution
- Sweep and Prune
  - Incremental
  - Full sort (radix/quicksort)
Uniform Grid

- Can be parallelized on GPU
- What cell size to use?
Voxelize objects

- Performance versus Precision
Sweep and Prune (SAP)

- Incremental version has data dependencies
- Sometimes, the best serial algorithm has poor parallel scalability [Intel07]
Parallel sorted SAP

- Array-based sorting [CE2005]
- Split each axis array into segments
- Sort segments in parallel and merge
- Traverse array to find all overlapping pairs
Parallel sorted SAP

- Need method to check overlap on other axis
  - Perform full AABB check
  - Compare array-indices instead of begin/endpoint values
Pair management

- Broadphase either:
  1) Rebuild all pairs from scratch
  2) Incrementally add/remove pairs
Adding/removing pairs

1) Use hash map to find existing pairs
2) Index into uniform grid (GPU)
3) Avoid random access find
Sort and cull pairs

- Perform AABB overlap check during traversal
- Mark non-overlapping pairs zero
- Look forward for duplicates
Pairwise Collision Dispatch

- Collision pairs are independent: embarrassingly parallel
- More culling needed for complex objects: AABB trees
- Output: contact points
Load balancing collision detection

- Broadphase is producer, mid/narrowphase multiple consumers
- Evenly distribute work, avoid synchronization
- Consume pairs while they are produced
Exceptions

- Deal with user callbacks
  - Collision filter callback (needs collision yes/no?)
- Serial fallback for unhandled cases (PPU)
Concave Shapes: AABB trees

- Make big tree fit cache/SPU
- Avoid random access due to recursion
- Output can go to narrow phase collision detector
Tree to array mapping

- Streaming large trees in parts
- Stackless traversal [EG2007]
Recursive versus Stackless

```c
void recurseTree(Node* node, Aabb* aabb) {
    if (testOverlap(node,aabb)) {
        if (isLeafNode(node))
            doNarrowPhase(node);
        else {
            recurseTree(node->leftChildNode);
            recurseTree(node->rightChildNode);
        }
    }
}

void iterateTree(Node* node, Aabb* aabb) {
    int curIndex=0;
    while (curIndex < endNodeIndex) {
        aabbOverlap = testOverlap(node,aabb);
        if (aabbOverlap )
            if (isLeafNode(node))
                doNarrowPhase(node);
        node += node->escapeIndex;
        curIndex += node->escapeIndex;
    }
}
```
Narrowphase collision algorithms

- Code size can become an issue for SPUs
- Can load/unload code using overlays or SPU Shaders [MA2008]
- Or use compact general purpose algorithms
Convex: GJK and EPA
GJK and EPA on SPUs

- See Gino van den Bergen’s book [Ber2003]
- Virtual methods like getSupportingVertex
  - Refactor into switch statement
- EPA polytope expansion can exceed memory
  - Use PPU fallback or other solution like SAT or Minkowski sampling
Contact Point Allocation

- Incremental or from scratch each frame
- Memory pre/de/re/allocation can be a big deal
- We preallocate fixed-size cache (4 points) for each pair
SPU & Contact Point Caching

- Contact cache is input and output
- Find/update existing points
- Perform contact reduction (to 4 points) on SPU
Solver parallelization

- Constraint solving methods using large matrices won’t fit
- Use iterative methods [Catto2005]
  - Sequential Impulse or Projected Gauss Seidel
- Constraints share rigidbodies: not embarrassingly parallel
Each island in parallel

- Works well, islands are independent
- Doesn’t parallelize large islands, load imbalance
- Overhead for small, single body islands
Grouping based on spatial hash

- Calculate cell dependency table
- Process cells in parallel, using table to avoid conflicts
Spatial hash function

//compute hash bucket index in [0,NUM_BUCKETS-1]
int32 getCellIndex (int32 cellpos_x, int32 cellpos_y, int32 cellpos_z)
{
    const int32 h1 = 0x8da6b343; //Large primes
    const int32 h2 = 0xd8163841;
    const int32 h3 = 0xcb1ab31f;
    int32 n = h1*cellpos_x + h2*cellpos_y + h3*cellpos_z;
    n = n%NUM_BUCKETS;
    if (n<0) n+=NUM_BUCKETS;
    return n;
}
//Example from Christer Ericson’s book ”Real-time collision detection”. 
Reordering constraints [Intel07]
Insomniac vs Bullet SPU Physics

SPU physics pipeline comparison presentation is available online from Insomniac Tech website [IN2008]
References

Books

- [TM2005] Patterns for Parallel Programming, Timothy Mattson et.al. Addison Wesley
- [Ber2003] Collision detection in interactive 3D environments, Gino van den Bergen, Morgan Kaufmann
- [PBA2005] Physics Based Animation, Kenny Erleben et. al. Charles River Media, page 613
Bullet physics library

- An open source 3D physics engine
- [http://bulletphysics.com](http://bulletphysics.com)
- Written in C++
- Parallelized for SPU and other architectures
- Check out SCEA booth #5905 at GDC Expo
  - SPU optimized Game Physics
- Licensed PlayStation 3 developers can request access to SPURS version